60 在数据库里，哪些操作会导致在表级别加锁呢？

在数据库里，哪些操作会导致在表级别加锁呢？

之前我们已经给大家讲解了数据库里的行锁的概念，其实还是比较简单，容易理解的，因为在讲解锁这个概念之前，对于多事务并发以及隔离，我们已经深入讲解过了，所以大家应该很容易在脑子里有一个多事务并发执行的概念。

在多个事务并发更新数据的时候，都是要在行级别加独占锁的，这就是行锁，独占锁都是互斥的，所以不可能发生脏写问题，一个事务提交了才会释放自己的独占锁，唤醒下一个事务执行。

如果你此时去读取别的事务在更新的数据，有两种可能：

- 第一种可能是基于mvcc机制进行事务隔离，读取快照版本，这是比较常见的；
- 第二种可能是查询的同时基于特殊语法去加独占锁或者共享锁。

如果你查询的时候加独占锁，那么跟其他更新数据的事务加的独占锁都是互斥的；如果你查询的时候加共享锁，那么跟其他查询加的共享锁是不互斥的，但是跟其他事务更新数据就加的独占锁是互斥的，跟其他查询加的独占锁也是互斥的。

当然一般我个人从多年研发经验而言，不是太建议在数据库粒度去通过行锁实现复杂的业务锁机制，而更加建议通过redis、zookeeper来用分布式锁实现复杂业务下的锁机制，其实更为合适一些。

为什么呢？因为如果你把分布式系统里的复杂业务的一些锁机制依托数据库查询的时候，在SQL语句里加共享锁或者独占锁，会导致这个加锁逻辑隐藏在SQL语句里，在你的Java业务系统层面其实是非常的不好维护的，所以一般是不建议这么做的。

比较正常的情况而言，其实还是多个事务并发运行更新一条数据，默认加独占锁互斥，同时其他事务读取基于mvcc机制进行快照版本读，实现事务隔离。

今天我们要给大家在讲完行锁之后，继续讲一个新的概念，就是表级锁。

在数据库里，你不光可以通过查询中的特殊语法加行锁，比如lock in share mode、for update等等，还可以通过一些方式在表级别去加锁。

有些人可能会以为当你执行增删改的时候默认加行锁，然后执行DDL语句的时候，比如alter table之类的语句，会默认在表级别加表锁。这么说也不太正确，但是也有一定的道理，因为确实你执行DDL的时候，会阻塞所有增删改操作；执行增删改的时候，会阻塞DDL操作。

但这是通过MySQL通用的元数据锁实现的，也就是Metadata Locks，但这还不是表锁的概念。因为表锁其实是InnoDB存储引擎的概念，InnoDB存储引擎提供了自己的表级锁，跟这里DDL语句用的元数据锁还不是一个概念。

只不过DDL语句和增删改操作，确实是互斥的，大家要知道这一点。

今天讲到这里，其实就是先给大家提一下表级锁的概念，同时梳理清楚DDL之类的语句是跟增删改操作互斥的，大家先理解到这个点就好

下一讲我们继续聊表级锁这个概念，说一下具体如何加锁，表级锁之间是如何互斥的。

End